

Efficient Analog Circuits for Boolean Satisfiability

Xunzhao Yin, Behnam Sedighi, Melinda Varga,
Mária Ercsey-Ravasz, Zoltán Toroczkai, Xiaobo Sharon Hu, *Fellow, IEEE*.

Abstract—Efficient solutions to NP-complete problems would significantly benefit both science and industry. However, such problems are intractable on digital computers based on the von Neumann architecture, thus creating the need for alternative solutions to tackle such problems. Recently, a deterministic, continuous-time dynamical system (CTDS) was proposed [14] to solve a representative NP-complete problem, Boolean Satisfiability (SAT). This solver shows polynomial analog time-complexity on even the hardest benchmark k -SAT ($k \geq 3$) formulas, but at an energy cost through exponentially driven auxiliary variables. With some modifications to the CTDS equations, here we present a novel analog hardware SAT solver, AC-SAT, implementing the CTDS. AC-SAT is intended to be used as a co-processor, and with its modular design can be readily extended to different problem sizes. The circuit is designed and simulated based on a 32nm CMOS technology. SPICE simulation results show speedup factors of $\sim 10^4$ on even the hardest 3-SAT problems, when compared with a state-of-the-art SAT solver on digital computers. For example, for hard problems with $N = 50$ variables and $M = 212$ clauses, solutions are found within from a few ns to a few hundred ns with an average power consumption of 130 mW .

I. INTRODUCTION

With Moore’s Law coming to end [35], exploring novel computational paradigms (e.g., quantum computing and neuromorphic computing) is more imperative than ever. While quantum computing is a promising venue, it is far from being brought to practical reality, with many challenges still to be faced, both in physics and engineering. Neuromorphic computing systems, e.g., Cellular Neural Networks (CNNs) [6], [7] and IBM’s TrueNorth [22], have been shown to be promising alternatives for solving a range of problems in, say, sensory processing (vision, pattern recognition) and robotics. Analog mix-signal information processing systems such as CNNs can offer extremely power/energy efficient solutions to some problems that are costly to solve by digital computers [30]. Such systems have received increasing attention in recent years (e.g., [8], [21], [32]).

In analog computing [20], the algorithm (representing the “software”) is a dynamical system often expressed in the form of differential equations running in continuous time over real numbers, and its physical implementation (the “hardware”) is any physical system, such as an analog circuit, whose behavior

is described by the corresponding dynamical system. The equations of the dynamical system are designed such that the solutions to problems appear as attractors for the dynamics and the output of the computation is the set of convergent states to those attractors [3]. Although it has been shown that systems of ordinary differential equations can simulate any Turing machine [4], [6], [29], and hence they are computationally universal, they have not yet gained widespread popularity due to the fact that designing such systems is problem specific and usually difficult. However, if an efficient analog engine can be designed to solve NP-complete problems, then according to the Cook-Levin theorem [16], it would help solve efficiently *all* problems in the NP class, as well as benefit a very large number of applications, both in science and engineering.

In this paper, we consider designing analog circuits for solving a representative NP-complete problem, the Boolean satisfiability (SAT) problem. SAT is quintessential to many electronic design automation problems, and is also at the heart of many decision, scheduling, error-correction and security applications. Here we focus on k -SAT, for which it is well known that for $k \geq 3$, k -SAT is NP-complete [16]. The currently best known deterministic, sequential discrete algorithm that exploits some properties of the search space has a worst case complexity of $\mathcal{O}(1.473^N)$ [5]. Other algorithms are based on heuristics and while they may perform well on some SAT formula classes, there are always other formulas on which they take exponentially long times or get stuck indefinitely. Some of the better known SAT solvers include Zchaff [24], MiniSat [13], RSat [26], WalkSAT [28], Focused Record-to-Record Travel (FRRT) [12] and Focused Metropolis Search (FMS) [27]. They typically consist of decision, deduction, conflict analysis and other functions [11] that employ the capability of digital computers to assign values to literals, conduct Boolean constraint propagation (BCP) and backtrack conflicts [31], [36].

A number of hardware based SAT solvers have been proposed in the past. FPGAs based solutions have been investigated to accelerate the BCP part found in all “chaff-like” modern SAT solvers [9], [10], [34]. Speedups of anywhere between 3X and 38X have been reported when comparing these FPGA based solvers over MiniSat [13], a well known, high-performance software solver. A custom digital integrated circuit (IC) based SAT solver, which implements a variant of general responsibility assignment software patterns (GRASP) and accelerates traversal of the implication graph and conflict clause generation, has been introduced in [17], [18]. A speed up of $\sim 10^3$ over MiniSat was reported based on simulation together with extrapolation. Performance of these hardware based approaches still have a lot of room for improvement since the algorithms that these hardware accelerators are based

X. Yin and X.S. Hu are with the Department of Computer Science and Engineering, University of Notre Dame, IN 46556, USA (xyin1, shu@nd.edu).

B. Sedighi is with Qualcomm, San Diego, CA 92121, USA (behnam.sedighi@gmail.com).

M. Varga and Z. Toroczkai are with the Department of Physics and the Interdisciplinary Center for Network Science and Applications (iCeNSA), University of Notre Dame, Notre Dame, IN, 46556 USA (mvarga, toro@nd.edu).

M. Ercsey-Ravasz is with the Faculty of Physics, Hungarian Physics Institute, Babes-Bolyai University, Cluj-Napoca, Romania, (ercsey.ravasz@phys.ubbcluj.ro)

on are designed for digital computers and thus can typically expect to achieve limited speedup.

Ref. [25] proposes a distributed mixed (analog and digital) algorithm that is implementable on VLSI devices. The algorithm is based on a heuristic method combined with stochastic search drawing on the natural incommensurability of analog oscillators. However, assuming $P \neq NP$, in order to have polynomially scaling solution times, one would require exponentially many computing elements, that is, exponentially scaling hardware resources. In contrast, our method [14] trades time-cost for *energy-cost*, which in practical terms may be preferable to massive amounts of hardware resources. It is quite possible that from an engineering point of view the ideal approach combines both types of tradeoffs: time vs energy and time vs hardware (distributed). The heuristic stochastic search in [25] is effectively a simulated annealing method, which implies high exponential runtimes for worst case formulas. In contrast, our analog approach is fully deterministic and extracts maximum information about the solution, embedded implicitly within the system of clauses and can solve efficiently the hardest benchmark SAT problems - at an energetic cost [14].

Recently, an analog SAT solver circuit was introduced in [2] using the theoretical proposal from [23] based on the CNN architecture. However, the theory in [23] has exponential analog-time complexity, and thus is much less efficient than the SAT solver from [14], which forms the basis for this paper. Furthermore, the circuit from [2] seems to have been implemented only for a 4×4 problem size, and no hardware simulation and comparison results were reported.

We propose a novel analog circuit hardware SAT solver, referred to as AC-SAT¹.

AC-SAT is based on a deterministic, continuous-time dynamical system (CTDS) in the form of coupled ordinary differential equations [14]. This system finds SAT solutions in analog polynomial time, however, at the expense of auxiliary variables that can grow exponentially, when needed (see Refs [14], [15] for details). Note that this CTDS is an incomplete solver, but it does minimize the number of unsatisfied clauses when there are no solutions, and thus it is also a MaxSAT solver. We present a *modular, programmable* implementation for this CTDS and validate our design through SPICE simulation. Our simulation results show that AC-SAT can significantly outperform (over tens of thousands times faster than) MiniSat with the latter running on a high-performance digital processor. For hard SAT problems with 50 variables and over 200 clauses, compared with the projected performance of a possible custom hardware implementation based a recent FPGA solver [34], AC-SAT offers more than $\sim 600X$ speedup.

In the rest of the paper, we first review the basic CTDS theory and some of its variants in Section II. Section III introduces the overall AC-SAT design. In Sec. III-C, we present two alternative designs for a specific component in AC-SAT. Sec. IV first discusses simulation-based validation of AC-SAT and compares the different component designs,

and then summarizes performance results for AC-SAT with respect to a software implementation of the CTDS SAT solver and MiniSat. Finally, we conclude the paper in Section V.

II. BACKGROUND

Solving a k -SAT problem is to find an assignment to N Boolean variables $x_i \in \{0, 1\}$, $i = 1, \dots, N$, such that they satisfy a given propositional formula F . F in conjunctive normal form (CNF) is expressed as the conjunction of M clauses C_m , $m = 1, \dots, M$, i.e., $F = \bigwedge_{m=1}^M C_m$, where each clause is formed by the disjunction of k literals (which are variables or their complements). An example of a clause for 3-SAT would be $C_5 = (x_3 \vee \bar{x}_{19} \vee x_{53})$. Following [14], an analog variable s_i , which can take any real value in the range $s_i \in [-1, 1]$, is associated with the Boolean variable x_i such that $s_i = -1$ corresponds to x_i being FALSE ($x_i = 0$) and $s_i = 1$ to x_i being TRUE ($x_i = 1$). The formula $F = \bigwedge_{m=1}^M C_m$ can be encoded via the $M \times N$ matrix $\mathbf{C} = \{c_{m,i}\}$ with $c_{m,i} = 1$ when x_i appears in clause C_m , $c_{m,i} = -1$ when its complement \bar{x}_i (negation of x_i) appears in C_m and $c_{m,i} = 0$ when neither appears in C_m . To every clause C_m , we associate an analog function $K_m(\mathbf{s}) \in [0, 1]$ given by

$$K_m(\mathbf{s}) = 2^{-k} \prod_{i=1}^N (1 - c_{m,i} s_i). \quad (1)$$

It is easy to see that clause C_m is satisfied, iff $K_m = 0$. Defining a “potential energy” function

$$V(\mathbf{s}, \mathbf{a}) = \sum_{m=1}^M a_m K_m^2, \quad (2)$$

where $a_m > 0$ are auxiliary variables, it is clear that all the clauses are satisfied iff $V = 0$. Thus the SAT problem can be reformulated as search in \mathbf{s} for the global minima of V (since the condition $V \geq 0$ always applies). If the auxiliary variables a_m are kept as constants, then for most hard problems any hill-descending deterministic algorithm (which evolves the variables $s_i(t)$) would eventually become stuck in local minima of V and not find solutions. To avoid this, the auxiliary variables are endowed with a time-dependence coupled to the analog clause functions K_m . Ref [14] proposed

$$\dot{s}_i = \frac{ds_i}{dt} = -\frac{\partial}{\partial s_i} V(\mathbf{s}, \mathbf{a}), \quad i = 1, \dots, N \quad (3)$$

$$\dot{a}_m = \frac{da_m}{dt} = a_m K_m, \quad m = 1, \dots, M \quad (4)$$

in which (3) describes a gradient descent on V and (4) is an exponential growth driven by the level of non-satisfiability in K_m (which also guarantees that $a_m(t) > 0$, at all times). (3) can be rewritten as

$$\frac{ds_i}{dt} = \sum_{m=1}^M a_m D_{m,i} \quad (5)$$

¹We refer to AC-SAT as an Analog Circuit SAT solver since its main processing engine is analog. However, the entire system is a mixed-signal one as a digital verification component is also included in the hardware system.

where

$$D_{m,i} = -\frac{\partial}{\partial s_i} K_m^2 = 2K_m c_{m,i} \prod_{\substack{j=1 \\ j \neq i}}^N (1 - c_{m,j} s_j). \quad (6)$$

For the auxiliary variables a_m , the formal solution to (4) is

$$a_m(t) = a_m(0) e^{\int_0^t d\tau K_m(s(\tau))}, \quad (7)$$

and thus the expression (2) of V is dominated by those K_m terms which have been unsatisfied for the longest time during the dynamics, resulting in an analog version of a focused search-type [27] dynamics. Also note that system (3) - (4) is not unique, however, it is simple from a theoretical point of view, and incorporates the necessary ingredients for solving arbitrary SAT problems, due to the exponentially accelerated auxiliary variables. For details on the performance of the algorithm see [14].

It is important to observe that while the scaling of the analog time t to find solutions is polynomial, in hardware implementations, the a_m variables represent voltages or currents and thus the energetic resources needed to find solutions may become exponential for hard formulas which is, of course necessary, assuming $P \neq NP$. However, the a_m variables do not need to grow exponentially all the time and unlimitedly, as in (4) and for that reason form (4) is not ideal for physical implementations. The challenge is then finding other variants that still significantly outperform digital algorithms, yet they are feasible in terms of physical implementations and costs. Note that such systems as ours essentially convert time costs into energy costs.

Here we introduce another form, based on time-delayed information, which keeps the focused search nature but allows the a_m 's to *decrease* as well when the corresponding clauses are (nearly) satisfied:

$$\frac{da_m}{dt} = a_m \{ K_m(s(t)) - [1 - \dot{\delta}_m(t)] K_m(s(t - \delta_m(t))) \}, \forall m \quad (8)$$

and we demand that

$$a_m(0) > 0, \text{ and } \delta_m(0) = 0, \forall m \quad (9)$$

where the delay functions $\delta_m(t) \in [0, t]$ determine the history window of $K_m(s(t))$ trajectory that has impact on the variation of a_m . The formal solution to (8) is:

$$a_m(t) = a_m(0) e^{\int_{t-\delta_m(t)}^t d\tau K_m(s(\tau))}. \quad (10)$$

Clearly, the case $\delta_m(t) = t$ corresponds to (4), while $\delta_m(t) = 0$ recovers the case of constant a_m 's which corresponds to the naive energy minimization case. One approach to choosing $\delta_m(t)$ is setting it to a small value initially and doubling it every time the dynamics is stuck or hits an upper threshold (set, e.g., by a maximum allowed voltage value). This typically only requires a few iterations. Other delay functions are being investigated. It is important to note that the decrease of satisfied clause's associated a_m due to this time-delayed form relatively reduces the clause's weight in (5), thus increases other clauses' weights in the focused search space, enhancing

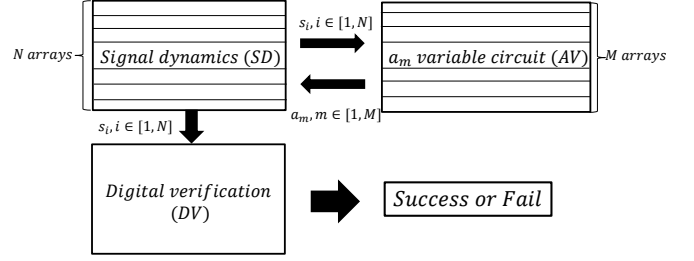


Fig. 1. High-level framework of AC-SAT. For a k -SAT problem, the SDC contains N elements corresponding to the N variables while the AVC contains M elements corresponding to the M clauses.

the driving capability of unsatisfied clauses. Next we present several analog circuit designs that implement the above equations, with some variations.

III. SYSTEM DESIGN

Fig. 1 depicts the high-level framework of AC-SAT. It consists of three main components: signal dynamics circuit (SDC), auxiliary variable circuit (AVC), and digital verification circuit (DVC). The AVC contains M identical elements, each of which receives the relevant s_i 's signals from the SDC as inputs, and generates a_m ($m \in [1, M]$) variables as outputs. The SDC, containing N identical elements, in turn receives a_m 's as feedback from the AVC and evolves the s_i ($i \in [1, N]$) signals accordingly. The SDC outputs the analog values of s_i 's to the AVC and the digital version of s_i 's to the DVC. Based on the digital values of s_i 's, the DVC determines whether a solution to the given SAT problem has been found at that time. The given framework can solve any k -SAT problems with N or less Boolean variables and M or less clauses. Below, we elaborate the design of the three circuit components using the 3-SAT problem (i.e., three non-zero $c_{m,j}$'s for each clause) as an example. AC-SAT for any k -SAT problem can be designed following the same principle.

A. Signal Dynamics Circuit

The SDC contains an array of analog elements that realize the dynamics specified by (5) and (6). Though it is possible to implement the multiplications and additions in (5) and (6) digitally, the hardware would be costly both in terms of area and power consumption and less efficient (see Sec. IV). We aim to develop an analog version of the dynamical system to achieve higher performance, smaller area and lower power consumption. We will show that the accuracy of the circuit is sufficient for the type of dynamical systems being considered here.

Given a 3-SAT problem with N variables, the SDC contains an array of N analog elements, referred to as s_i element, for evaluating the s_i ($i = 1, \dots, N$) signals. Fig. 2(a) shows the conceptual design of the s_i element that realizes (5). The s_i element contains a capacitor, C , connected to the M Branch blocks (where M is the total number of clauses in the 3-SAT problem), an analog inverter, an inverted Schmitt trigger and a digital inverter. The voltage across capacitor C , i.e., V_i , and the output of the analog inverter, \bar{V}_i , represent the analog value of signal s_i and $-s_i$, respectively. Signal $s_i \in$

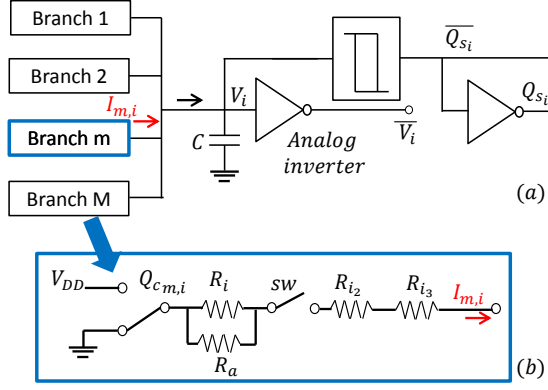


Fig. 2. (a) Design of one array element in the SDC. (b) Detailed design of the Branch block.

$[-1, 1]$ (resp., $-s_i \in [1, -1]$) is mapped to $V_i \in [GND, V_{DD}]$ (resp., $\bar{V}_i \in [V_{DD}, GND]$). The inverted Schmitt trigger and the digital inverter output the digital versions of $-s_i$ and s_i , denoted by \bar{Q}_{s_i} and Q_{s_i} , (i.e., taking on values of either GND or V_{DD}).

To see why the s_i element in Fig. 2(a) can be used to evaluate (5), let us denote the current from each of the Branch block as $I_{m,i}$. Then we have

$$C \frac{dV_i}{dt} = \sum_{m=1}^M I_{m,i} \quad (11)$$

Comparing (5) with (11), we see that the s_i element in Fig. 2(a) precisely realizes (5) if we have

$$I_{m,i} = C a_m D_{m,i} \quad (12)$$

In order to design a Branch block to satisfy (12), we first briefly discuss some observations associated with $D_{m,i}$. In a 3-SAT problem, there are only three non-zero $c_{m,j}$'s in each C_m clause. Let us denote them as $c_{m,i}$, $c_{m,i2}$, and $c_{m,i3}$. Then $D_{m,i}$ in (5) can be shown to have the following form:

$$D_{m,i} = 2^{-2} \times \begin{cases} c_{m,i}(1 - c_{m,i}s_i)(1 - c_{m,i2}s_{i2})^2(1 - c_{m,i3}s_{i3})^2 = \\ 2^{-2}(+1 - s_i)(1 - c_{m,i2}s_{i2})^2(1 - c_{m,i3}s_{i3})^2 & \text{if } c_{m,i} = 1 \\ 0 & \text{if } c_{m,i} = 0 \\ 2^{-2}(-1 - s_i)(1 - c_{m,i2}s_{i2})^2(1 - c_{m,i3}s_{i3})^2 & \text{if } c_{m,i} = -1 \end{cases} \quad (13)$$

Referring to (13), one can readily see that $D_{m,i}$ has the following properties:

- If any of s_i , s_{i2} and s_{i3} is satisfied, i.e., reaches 1 or -1 (indicating x_i is either TRUE or FALSE), $D_{m,i}$ becomes zero. According to (5), a zero $D_{m,i}$ means that clause C_m has no impact on the variation of s_i . On the other hand, when none of the three variables is satisfied, but one of them gets closer to being satisfied, the magnitude of $D_{m,i}$ reduces, again.
- The sign of $D_{m,i}$ is the same as that of $c_{m,i}$ since $(1 - s_i)$ cannot be negative. If $c_{m,i} = 1$ (resp., -1), the contribution of $D_{m,i}$ to $\frac{ds_i}{dt}$ is positive (resp., negative), i.e., it tries to push s_i toward $+1$ (resp., -1).

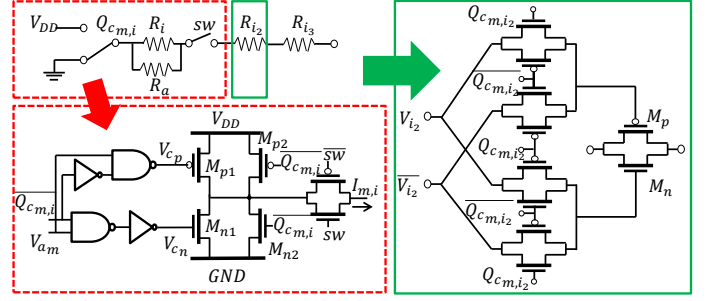


Fig. 3. Actual circuit implementation of the conceptual Branch block design in Fig. 2(b). The red box (lower part) includes the implementation of the switch as well as R_a and R_i . The green box (on the right) includes the implementation for R_{i2} and R_{i3} .

Based on the above observations, let us examine the conceptual design of the Branch block in Fig. 2(b). Specifically, the Branch block contains two switches and four tunable resistive elements. (We use resistors here to simplify the drawing, the details of these resistive elements will be given later.) One switch is controlled by signal sw , which is left open if $c_{m,i} = 0$ (indicating that x_i does not appear in clause C_m), and is closed otherwise. The other switch is controlled by $Q_{c_{m,i}}$, the digital version of $c_{m,i}$. If $c_{m,i} = 1$ (resp., -1), indicating that x_i (resp., \bar{x}_i) is present in the clause, the switch controlled by $Q_{c_{m,i}}$ connects to V_{DD} (resp., GND). It can be readily seen that

$$I_{m,i} = \begin{cases} (V_{DD} - V_i)/(R_{i2} + R_{i3}) & \text{if } c_{m,i} = 1 \\ 0 & \text{if } c_{m,i} = 0 \\ (GND - V_i)/(R_{i2} + R_{i3}) & \text{if } c_{m,i} = -1 \end{cases} \quad (14)$$

If the values of R_a , R_i , R_{i2} and R_{i3} are chosen properly, the $I_{m,i}$ value derived from the Branch block would have the same properties as identified for $D_{m,i}$ above.

The actual realization of the four resistive elements in Fig. 2(b) is given in Fig. 3. The implementation of R_{i2} and that of R_{i3} are the same and the one for R_{i2} is shown on the right of Fig. 3. Consider the R_{i2} block. The two terminals of the transmission gate formed by transistor M_{p1} and M_{n1} correspond to the terminals of R_{i2} . The gate terminals of M_{p1} and M_{n1} are connected to V_{i2} and \bar{V}_{i2} via four additional transmission gates controlled by $Q_{c_{m,i2}}$ and $\bar{Q}_{c_{m,i2}}$. It can be derived that this realization of R_{i2} exhibits the desired properties outlined above for $D_{m,i}$. For example, assuming that $c_{m,i2}$ is 1, i.e. $Q_{c_{m,i2}} = V_{DD}$, the gates of M_{n1} and M_{p1} are connected to \bar{V}_{i2} and V_{i2} , respectively. If variable s_{i2} is satisfied, i.e. V_{i2} is close to V_{DD} and \bar{V}_{i2} is close to GND , then both M_{n1} and M_{p1} are OFF, R_{i2} has a very large value and $I_{m,i}$ becomes zero. This means that clause C_m has no impact on the variation of s_i which is exactly the desired behavior. On the other hand, if s_{i2} is not satisfied, as it gets closer to its target (either $+1$ or -1), the magnitude of $I_{m,i}$ reduces because R_{i2} increases.

The circuit block for implementing the switch controlled by $Q_{c_{m,i}}$ and the two resistive elements R_i and R_a is shown in the red box on the lower left in Fig. 3. The gates of transistor M_{p1} and M_{p2} (resp., M_{n1} and M_{n2}) control the connection to V_{DD} (resp., GND). One of the gates connects directly to

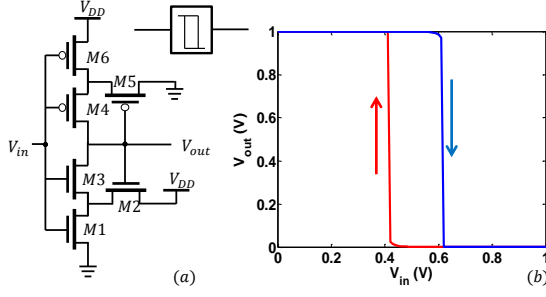


Fig. 4. Schmitt-trigger inverter and its transfer characteristic.

$\overline{Q_{c_{m,i}}}$ (representing the negated $c_{m,i}$ signal) while the other is controlled by

$$V_{c_p} = \overline{V_{a_m} Q_{c_{m,i}}} \quad \text{resp.,} \quad V_{c_n} = V_{a_m} \overline{Q_{c_{m,i}}} \quad (15)$$

Note that though V_{a_m} in (15) (as input to the two NAND gates) seems to be treated as a digital signal, it actually remains as an analog signal while the NAND gates and inverters operate in the linear $V_{in} - V_{out}$ region to produce analog outputs as desired.

It can be readily shown that the block realizes the switch function due to $c_{m,i}$ as well as $(+1 - s_i)$ and $(-1 - s_i)$ in (13), and incorporates the a_m term in (5). If $c_{m,i} = -1$, i.e. $Q_{c_{m,i}} = 0$, $V_{c_p} = V_{DD}$ and $V_{c_n} = V_{a_m}$, which means that the Branch block is connected to GND and the current flowing through the block is dependent on the voltage representing a_m , V_{a_m} . As V_{a_m} gets larger, M_{n1} exhibits smaller resistance, resulting in larger impact of a_m on the current flowing through the Branch block. Note that initially V_{a_m} is very small, thus the right two transistors M_{p2} and M_{n2} which are of smaller sizes than M_{p1} and M_{n1} are employed to ensure proper current flow as well as serve as a current boost.

The SDC also converts the analog signals, V_i 's, to digital signals, Q_{s_i} 's, via an inverted Schmitt trigger. The inverted Schmitt trigger circuit is shown in Fig. 4(a). The digital signals are then sent to the DVC to check if a solution has been found. The inverted Schmitt trigger circuit exhibits hysteresis in its transfer curve as seen from the simulation result in Fig. 4(b), hence can perform analog-digital conversion with minimal noise impact. Putting all the above discussions together, one can conclude that the SDC correctly implements the system dynamics defined by (3).

B. Auxiliary Variable and Digital Verification Circuits

As pointed out in Section II, the auxiliary variables, a_m 's as defined in (4), are used to help avoid the gradient descent search being stuck in non-solution attractors. The a_m signal follows an exponential growth driven by the level of non-satisfiability in clause C_m . A direct way to implement an exponential function is through an operational amplifier (op-amp), which we present below.

The AVC contains an array of M a_m elements where M is the number of clauses in the given problem. Fig. 5 illustrates the conceptual design of the a_m element, similar to a non-inverting integrator. Here, the value of a_m (for clause C_m) is represented by the voltage at the output of the op-amp, i.e.,

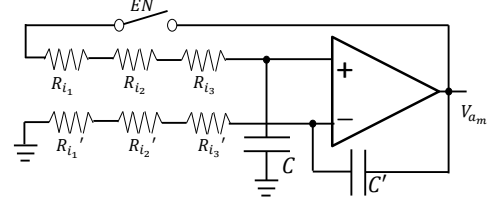


Fig. 5. Conceptual implementation of one element in the AVC. Actual realization of the resistive element is similar to the circuit in Fig. 3.

V_{a_m} . Resistive elements R_{i1} , R_{i2} and R_{i3} are identical to R'_{i1} , R'_{i2} and R'_{i3} , respectively, and the two capacitors, C and C' , have identical values as well. The switch controlled by EN is realized by a transmission gate to control the start of the a_m element. The first order differential equation of V_{a_m} can be written as:

$$C \frac{dV_{a_m}}{dt} = V_{a_m} / (R_{i1} + R_{i2} + R_{i3}). \quad (16)$$

The R 's in Fig. 5 are tunable resistive elements implemented by transmission gates similar to that shown in the green box of Fig. 3. For example, for R_{i1} and R'_{i1} , the transmission gates (M_p and M_n) are controlled, via four other transmission gates, by analog signals V_{i1} and V'_{i1} , representing x_i 's presence in clause C_m . The other two pairs of R 's are designed in the same way. If any of the three variables in clause C_m is satisfied, the corresponding V_i turns off the respective transmission gate and cut off the current paths from op-amp's inverting input to ground and from non-inverting input to V_{a_m} .

The circuit in Fig. 5 exactly realizes the exponential growth specified in (4), albeit with an upper bound on V_{a_m} , i.e. the op-amp's supply voltage. If V_{a_m} reaches its upper bound before any of the three variables in the corresponding clause is satisfied, the circuit stops evolving since the V_{a_m} value is unable to drive this yet unsatisfied clause any more. This impacts the effectiveness of avoiding being stuck in a non-solution attractor during the gradient descent search process. The upper bound on V_{a_m} imposes a physical limitation on the hardware realization of the CTDS theory².

Although the AVC design given in Fig. 5 realizes the exponential growth, it requires M op-amps, resulting in a large amount of area and power consumption. There exist other ways to achieve exponential signal growth, e.g., circuits with positive feedback often have exponential growth in certain ranges. Besides the exponential growth implementation, we will introduce alternative AVC designs in the next subsection.

The goal of the DVC is to determine if a solution (the set of s_i 's) to the given problem has been found. The DVC is implemented readily through the use of an array of $3M$ XOR gates and an array of M NAND gates as shown in Fig. 6. The input to the DVC is the digital representation of s_i 's and $-s_i$'s, i.e., Q_{s_i} and $\overline{Q_{s_i}}$, from the SDC. Each NAND gate corresponds

²As discussed in Sec. II, Eqs. (3)-(4) are not unique, and the effect of the maximum voltage limitation depends on the equations themselves. For example, Sec. II introduces an alternative, delay-based formulation for a_m in (8) which allows a_m to decrease when the corresponding clause is satisfied. This delay-based formulation of a_m postpones reaching the a_m upper bound. The implementation of (8) for the op-amp based approach is currently under development.

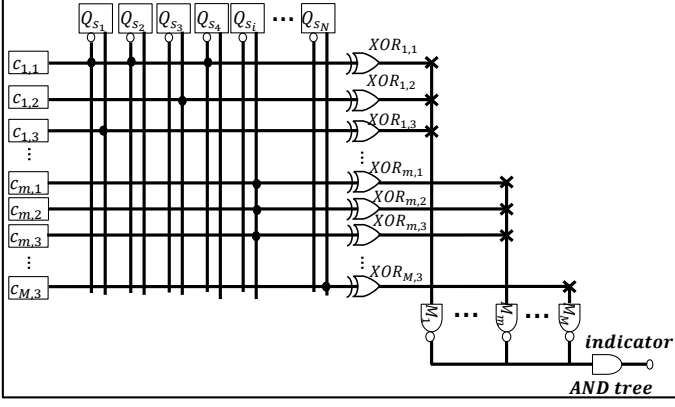


Fig. 6. Schematic of the DVC.

to a clause and its inputs correspond to the literals present in the clause. Note that in the DVC, we only include those $c_{m,i}$'s whose values are +1 (represented by logic signal "1") and -1 (represented by logic "0").

It is easy to see that all three components, SDC, AVC, and DVC, are modular and programmable. By modular, we mean that the basic elements in each circuit can be repeated for different problem sizes (i.e., number of variables, N , and number of clauses, M). By programmable, we mean that any k -SAT problem instance can be solved by the same SDC, AVC and DVC implementation as long as the problem size is less than or equal to the hardware specification.

C. Alternative AVC Designs

The op-amp based AVC described in Section III-B realizes an exponentially growing a_m variable aiming to address hard SAT problems (some SAT instances with constraint density $\alpha = M/N \gtrsim 4.25$) within its physical limitation. However, for application type SAT problems, i.e. which are not specially designed to be very hard, exponential growth for a_m is not always necessary. Below we describe two alternative circuit designs to implement an a_m function that has a $(1 - \epsilon_2 e^{-qt})$ -type growth to a saturation value. In the remainder we will refer to this version of a_m growth as the "simpler version".

Fig. 7 depicts the conceptual design of the a_m element realizing the simpler a_m growth, where capacitor C is charged to V_{DD} through three tunable resistors. The first order differential equation that governs V_{a_m} can be written as

$$C \frac{dV_{a_m}}{dt} = (V_{DD} - V_{a_m}) / (R_{i_1} + R_{i_2} + R_{i_3}). \quad (17)$$

R_{i_1} , R_{i_2} , R_{i_3} are same as the resistors in Fig. 5, realized by transmission gates controlled by V_{i_1} , V_{i_2} and V_{i_3} , similar to that in Fig. 3. If any of the three variables s_i in clause C_m is satisfied, the corresponding V_i turns off the respective transmission gate and cut off the current path from V_{DD} to the capacitor. This circuit guarantees the continuous growth of a_m since V_{a_m} is charged by V_{DD} till it reaches its upper bound V_{DD} or any of the three variables in the clause is satisfied.

It is important to note that as the circuit in Fig. 7 does not realize the exponential growth specified in (4), it can indeed get captured into non-solution attractors indefinitely for some

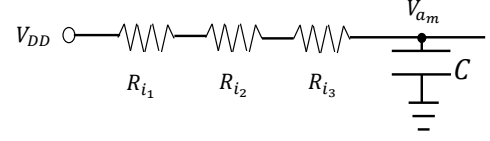


Fig. 7. Conceptual design of the AVC element realizing the $(1 - \epsilon_2 e^{-qt})$ -type growth. Implementation of the resistive elements is similar to those in Fig. 3.

very hard formulas. However, we have found that even for many hard problems, it works more efficiently than the op-amp based a_m element (with the same threshold value) in finding solutions for smaller size problems (as long as they are solvable), and the dynamics would only rarely get stuck. We will discuss this aspect more in the evaluation section via simulation results.

Similar to the op-amp based AVC, in the simpler AVC design in Fig. 7, some V_{a_m} 's may reach V_{DD} before the CTDS converges to a solution. One way to alleviate this physical limitation is to increase the range of V_{a_m} . However, such an approach has its limitations in practical circuits (e.g., the limited voltage supply allowed). This, in fact, is a fundamental limitation due to the NP hardness of 3-SAT. Nonetheless, it is possible to improve the V_{a_m} driving capability in the CTDS and increase the size of the hard problems that can be solved with the same physical range of V_{a_m} .

Below, we discuss an alternative implementation of the simpler a_m element to demonstrate that it is worthwhile to investigate different implementations of the AVC.

Recall that the delay function $\delta_m(t)$ in (8) is to assist a_m to keep relevant information from a limited range of the trajectory's past history instead of the entire history. We consider combining the simpler a_m element with this time-delayed form, and choose $\delta_m(t) = \delta$ (meaning that we are integrating over a fixed time window of length δ). The corresponding a_m element is shown in Fig. 8. Capacitor C is charged to V_{DD} through three tunable resistive elements and discharged to GND through the other three resistive elements. The first order differential equation of V_{a_m} can be written as

$$C \frac{dV_{a_m}}{dt} = \frac{V_{DD} - V_{a_m}}{R_{i_1} + R_{i_2} + R_{i_3}} + \frac{-V_{a_m}}{R'_{i_1} + R'_{i_2} + R'_{i_3}} \quad (18)$$

The six resistive elements are implemented by transmission gates similar to those for R_{i_2} in Fig. 3. Specifically, R'_{i_1} , R'_{i_2} , R'_{i_3} are controlled by s 's (representing all s_i 's) earlier values, i.e., $s(t - \delta_m)$. A chain of an odd number of analog inverters is used to realize the delay δ as shown at the right of Fig. 8. If signals $s(t)$ reach their targets at time t , the path from V_{DD} to capacitor C is cut off, while the discharge path is still conducting current. V_{a_m} keeps decreasing until the discharge path is cut off after δ . Hence the circuit properly implements the time-delayed simpler a_m growth function.

IV. EVALUATION

We have built our proposed analog SAT solver, AC-SAT, at the transistor level in HSPICE based on the PTM 32nm CMOS model [1]. All the circuit components use $V_{DD} = 1V$. To achieve sufficient driving capability, the minimum transistor size is set to $W = 1\mu m$, $L = 40nm$ while actual transistor

sizes are selected according to their specific roles. For logic gates, the transistor sizes are chosen to ensure equal pull-up and pull-down strength. For the Branch block in Fig. 3, the relative W/L values of transistor M_{p1} and M_{n1} (i.e., the size of R_a) with respect to the W/L values of the other transistors (i.e., the sizes of R_i , R_{i2} and R_{i3}) determine the contribution of a_m to $I_{m,i}$. Thus, tradeoff between the size of R_a and the impact of a_m should be carefully considered. In our implementation, since R_i (dependent on the size of M_{p2} and M_{n2}) is mainly used for proper current flow at the beginning, it should not dominate the current flow as R_a starts affecting $I_{m,i}$. Hence we chose the transistor sizes such that they result in the ratio of R_a to R_i , R_{i2} and R_{i3} being 64, 4 and 4, respectively. The sizes of the transistors in other circuits are determined in a similar fashion.

To demonstrate that AC-SAT indeed behaves as specified by the CTDS dynamics in (3) and (4), we examine the waveforms of signals s_i and a_m . Fig. 9 shows three sets of s_i and a_m waveforms from a 3-SAT problem instance having 50 variables and 212 clauses: Fig. 9(a) for the op-amp based a_m implementation (realizing the $(\epsilon_1 e^{qt})$ -type a_m growth), Fig. 9(b) for the simpler a_m implementation (realizing the $(1 - \epsilon_2 e^{-qt})$ -type a_m growth), and Fig. 9(c) for the time-delayed simpler a_m implementation. For all three designs, AC-SAT successfully finds a solution after a certain time as indicated by the vertical dashed lines. Note that AC-SAT determines whether a solution is found via the DVC. As can be seen from the s_i trajectories, the s_i signals stabilize (i.e., converge) after a solution is found. Comparing the a_m trajectories in the three different designs, one can see that the a_m 's grow most rapidly in the op-amp based design due to the exponential growth function while some of the a_m 's (the ones corresponding to the satisfied clauses) in the time-delayed implementation decrease after they reach their peak magnitude, just as predicted by (8).

Readers may notice that the time scale of the waveforms in Fig. 9(a) is different from that of the other two sets of waveforms. For this particular problem instance, the op-amp based implementation takes much longer time to find a solution than the other two designs. To get a better comparison between the different designs, we employ both the op-amp based and the simpler a_m based AC-SAT to solve 2000 randomly generated, hard ($\alpha = 4.25$) 3-SAT problems containing 1000 instances of a small problem size ($N=10$) and 1000 instances of a large problem size ($N=50$). It has been verified that all these 2000 instances are solvable. With the same fixed supply voltage

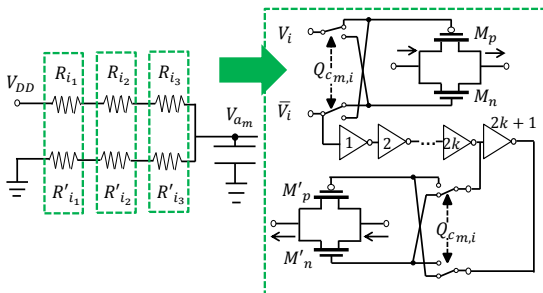


Fig. 8. Circuit realization of the time-delayed simpler a_m growth.

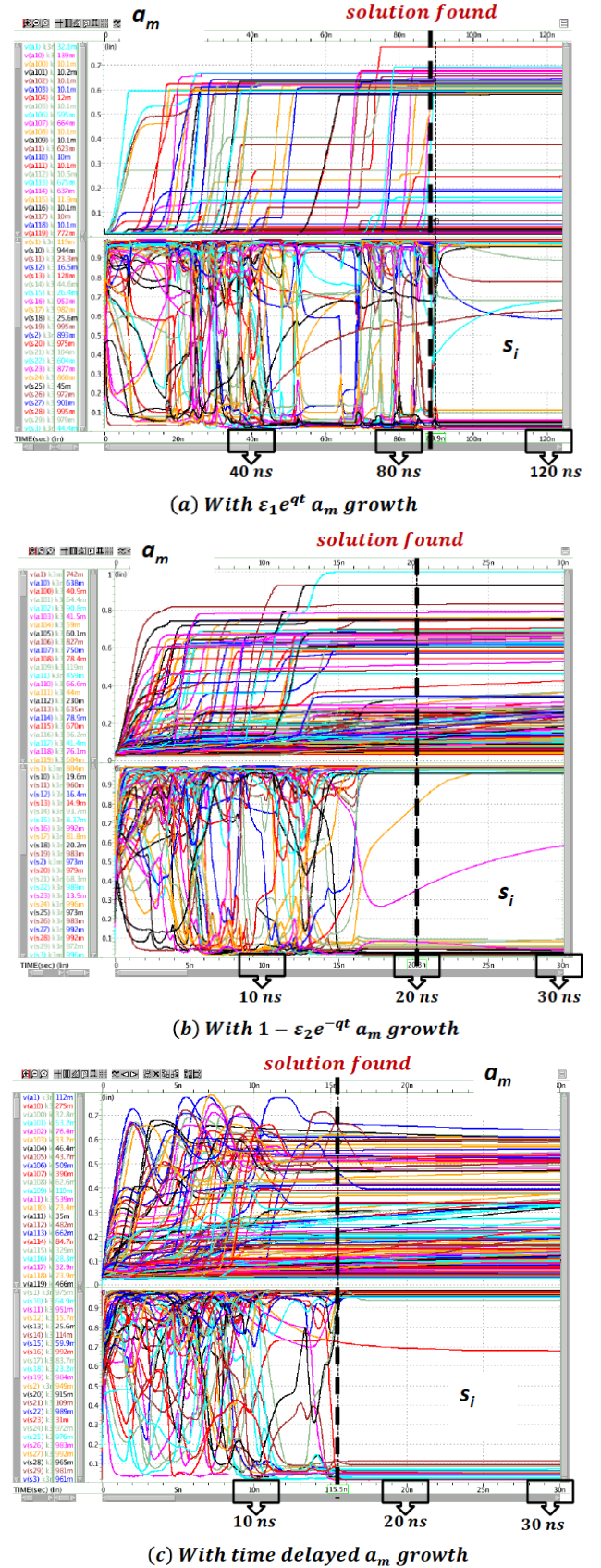


Fig. 9. Waveforms of signals representing $s_i(t)$ and $a_m(t)$ for a 3-SAT problem with 212 clauses and 50 signal variables. The problems/formulas have a constraint density $\alpha=M/N=4.25$, and are considered to be hard problems.

and initial conditions, AC-SAT with the op-amp based a_m design solves 70.9% of the N=10 instances and 52.0% of the N=50 instances, respectively, while AC-SAT with the simpler a_m design solves 84.9% of the N=10 instances and 39.0% of the N=50 instances, respectively. Note that consistent with the NP-complete nature of the problem, AC-SAT may not solve all problems because of the imposed physical voltage limit. However, allowing a higher voltage limit, the range of solved problems increases accordingly. These results indicate that, as expected, within the same physical constraints, AC-SAT based on the exponential growth a_m is better for larger size problems which have a higher chance of getting the circuit stuck, while AC-SAT with the $(1 - \epsilon_2 e^{-qt})$ -type a_m growth is more efficient in dealing with smaller size problems.

To further investigate the effectiveness of AC-SAT, we compare the simpler a_m based AC-SAT design with (i) a software program that solves the system (3)-(4) using an adaptive Runge-Kutta, fifth-order Cash-Karp method and (ii) the software MiniSat solver [13]. The software programs are running on the same digital computer. We randomly generated 5000 hard ($\alpha = 4.25$) 3-SAT problems that contain 1000 instances for each problem size of N=10, 20, 30, 40, 50. The same initial conditions are applied whenever appropriate. Table I summarizes the average time needed to find solutions for each problem size. The AC-SAT column reports the analog/physical times taken by AC-SAT. The CTDS and MiniSat columns report the CPU times of the two software implementations, respectively. (To be fair, only the times taken by the solved problems for all three methods are included.) Observe that the times in the CTDS column increase nearly exponentially as the problem size increases. This is natural, since the numerical integration happens on a digital Turing machine, and in order to ensure the pre-set accuracy of computing the chaotic trajectory the Runge-Kutta algorithm has to do a very large number of window-refining discretization steps. As seen from the data in Table I, AC-SAT demonstrates average speedup factors of $\sim 10^5$ to $\sim 10^6$ and $\sim 10^4$ over software CTDS and MiniSat, respectively.

AC-SAT is also very competitive compared with existing hardware based approaches. For example, a recent work [34] reported a CPU+FPGA based MiniSat solver achieving $\sim 4X$ performance improvement over CPU based MiniSat. Since ASIC implementations typically achieves a maximum of $10X$ performance improvement over their FPGA counterparts [19], compared with a projected ASIC version of the FPGA design in [34], AC-SAT would still result in $\sim 600X$ or higher speedup. We do not directly compare with the custom digital IC in [17] since our simulation-based system cannot solve the large size problems considered in [17]. (Note that the total solving times reported in [17] are extrapolated instead of directly obtained from simulation.) It is reported in [17] that an average speedup of $\sim 10^3X$ over CPU based MiniSat is obtained. As contrast, AC-SAT achieves $\sim 10^4X$ speedup over CPU based MiniSat. The average power consumption of AC-SAT for the size N=50 problems is 130 mW which is mostly due to the static power dissipation by the analog inverters and NAND gates.

Readers may be concerned with the complexity of the

TABLE I
PERFORMANCE COMPARISON OF AC-SAT, SOFTWARE CTDS AND MINISAT

SAT solver		AC-SAT	CTDS	MiniSat
Platform		ASIC 32nm CMOS	Intel Core i7-4700 @2.4 GHz	Intel Core i7-4700 @ 2.4GHz
Average time for each size N (s)	N=10	4×10^{-9}	4.40×10^{-4}	2.3×10^{-4}
	N=20	7×10^{-9}	3.91×10^{-3}	2.4×10^{-4}
	N=30	10^{-8}	1.62×10^{-2}	2.8×10^{-4}
	N=40	1.2×10^{-8}	5.22×10^{-2}	3.1×10^{-4}
	N=50	1.4×10^{-8}	1.13×10^{-1}	3.7×10^{-4}

analog hardware design as well as other issues such as noise. It is important to note that the analog solver core is modular and consists of arrays with the same topologies. Furthermore, the CTDS theory has been shown to be robust against noise [33]. AC-SAT can be easily made to be programmable and used as a co-processor such that the host processor sets up the $c_{m,i}$ values for a given problem instance, lets AC-SAT evolve and receives the solution from AC-SAT after the solution is found. The current implementation of AC-SAT does have some limitations. For example, though a circuit can handle any problems with a fixed k value, fewer number of variables and/or clauses once the circuit is designed, it cannot handle problems with larger number of variables or clauses as well as larger k value. We plan to address these challenges in our future work.

V. CONCLUSIONS

We presented a proof-of-principle analog system, AC-SAT, based on the CTDS in [14] to solve 3-SAT problems. The design can be readily extended to general SAT problems. AC-SAT is modular, programmable and can be used as a SAT solver co-processor. In this implementation the circuit size grows polynomially ($O(N^2)$) as the problem size increases. Three different design alternatives were proposed for implementing the auxiliary variable dynamics required by the CTDS. Detailed SPICE simulation results show that AC-SAT can achieve $\sim 10^4X$ speedup over MiniSat running on a state-of-the-art digital processor, and can offer over 600X speedup over projected digital ASIC implementation of MiniSat.

The CTDS equations (especially the dynamics for the auxiliary variables) and their analog implementations are not unique. It is quite possible that better forms and implementations exist. The fact that our proof-of-principle circuit implementations significantly outperform state-of-the-art solvers on digital computers are an indication that analog hardware solvers, as shown here for SAT, have a great potential as processors for discrete optimization. As future work, we are investigating alternative implementations of the auxiliary variable dynamics as well as methods to handle problem instances that do not fit on a given hardware implementation. Specific challenges associated with analog hardware such as noise and fabrication variations will also be further investigated.

REFERENCES

- [1] Predictive Technology Model (PTM). <http://ptm.asu.edu/>.
- [2] D. Basford, et al. The impact of analog computational error on an analog boolean satisfiability solver. In *ISCAS*, 2016.

- [3] A. Ben-Hur, et al. A theory of complexity for continuous time systems. *Journal of Complexity*, 18:51–86, 2002.
- [4] M. Branicky. Analog computation with continuous odes. *Workshop on Physics and Computation, Dallas TX USA*, pages 265–274, 1994.
- [5] T. Brueggemann et al. An improved local search algorithm for 3-sat. *Theoretical Computer Science*, 329(1-3):303–313, 2004.
- [6] L. Chua, et al. The cnn is universal as the turing machine. *TCAS I*, 40(4):289–291, APR 1993.
- [7] L. Chua et al. Cellular neural networks - theory. *TCAS I*, 35(10):1257–1272, OCT 1988.
- [8] R. Daniel, et al. Synthetic analog computation in living cells. *Nature*, 497(7451):619–623, 2013.
- [9] J. D. Davis, et al. Designing an efficient hardware implication accelerator for sat solving. In *SAT 2008*, pages 48–62. Springer, 2008.
- [10] J. D. Davis, et al. A practical reconfigurable hardware accelerator for boolean satisfiability solvers. In *DAC*, pages 780–785, 2008.
- [11] M. Davis, et al. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
- [12] G. Dueck. New optimization heuristics. *J. Comput. Phys.*, 104(1):86–92, 1993.
- [13] N. Eén et al. An extensible sat-solver. In *Theory and applications of satisfiability testing*, pages 502–518. Springer, 2003.
- [14] M. Ercsey-Ravasz et al. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nature Physics*, 7(12):966–970, DEC 2011.
- [15] M. Ercsey-Ravasz et al. The chaos within Sudoku. *Scientific Reports*, 2:725, OCT 11 2012.
- [16] M. R. Garey et al. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd, first edition edition, Jan. 1979.
- [17] K. Gulati et al. Accelerating boolean satisfiability on a custom ic. In *Hardware Acceleration of EDA Algorithms*, pages 33–61. Springer, 2010.
- [18] K. Gulati, et al. Efficient, scalable hardware engine for boolean satisfiability and unsatisfiable core extraction. *Computers & Digital Techniques, IET*, 2(3):214–229, 2008.
- [19] I. Kuon et al. Measuring the gap between fpgas and asics. *TCAD*, 26(2):203–215, 2007.
- [20] S.-C. Liu, et al. *Analog VLSI - Circuits and Principles*. MIT Press, November 2002.
- [21] J. Lu, et al. A 1 tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos. *ISSC*, 50(1):270–281, 2015.
- [22] P. A. Merolla, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.
- [23] B. Molnár et al. Asymmetric continuous-time neural networks without local traps for solving constraint satisfaction problems. *PloS one*, 8(9):e73400, 2013.
- [24] M. W. Moskewicz, et al. Chaff: Engineering an efficient sat solver. In *DAC*, pages 530–535. ACM, 2001.
- [25] H. Mostafa, et al. An event-based architecture for solving constraint satisfaction problems. *Nature Communications*, 6:8941, December 2015.
- [26] K. Pipatsrisawat et al. Rsat 2.0: Sat solver description. *SAT competition*, 7, 2007.
- [27] A. Seitz, et al. Focused local search for random 3-satisfiability. *J. of Statistical Mechanics: Theory and Experiment*, page P06006, 2005.
- [28] B. Selman, et al. Local search strategies for satisfiability testing. In *DIMACS Series*, pages 521–532, 1996.
- [29] H. Siegelmann. Computation beyond the turing limit. *Science*, 268:545–548, 1995.
- [30] H. Siegelmann. *Neural networks and analog computation: beyond the Turing limit*. Springer Science & Business Media, 2012.
- [31] J. P. M. Silva et al. Grasp—a new search algorithm for satisfiability. In *ICCAD*, pages 220–227. IEEE Computer Society, 1997.
- [32] R. St Amant, et al. General-purpose code acceleration with limited-precision analog computation. *ACM SIGARCH CAN*, 42(3):505–516, 2014.
- [33] R. Sumi, et al. Robust optimization with transiently chaotic dynamical systems. *EPL (Europhysics Letters)*, 106(4):40002, 2014.
- [34] J. Thong et al. Fpga acceleration of enhanced boolean constraint propagation for sat solvers. In *ICCAD*, pages 234–241. IEEE Press, 2013.
- [35] M. M. Waldrop. More than Moore. *Nature*, 530:144–147, February 2016.
- [36] L. Zhang, et al. Efficient conflict driven learning in a boolean satisfiability solver. In *ICCAD*, pages 279–285. IEEE Press, 2001.